

04

Method - Overload and Override



Xcode

2013.10
경기과학고등학교

- Method란 구조체에서 선언된 함수를 의미한다. 쉽게 말하면 구조체 내부함수라고 생각하면 된다.
- 구조체 이름과 동일한 이름의 Method를 생성자라고 한다.
- 구조체는 Overloading을 이용하여 다양한 방법으로 초기화 할 수 있다.
- 구조체로 선언된 변수들 끼리의 연산은 정의되어 있지 않다. 따라서 overriding를 이용하여 연산자를 재정의하여 사용할 수 있다.

시간을 나타내는 구조체 TIME

member : hour, min, sec

구조체 선언

```
struct TIME  
{  
    int hour, min, sec;  
};
```

- 전역 변수로 구조체를 정의한다.
- 각 member은 시간, 분, 초를 의미한다.

TIME의 기본적인 활용 (시간 입력 및 출력)

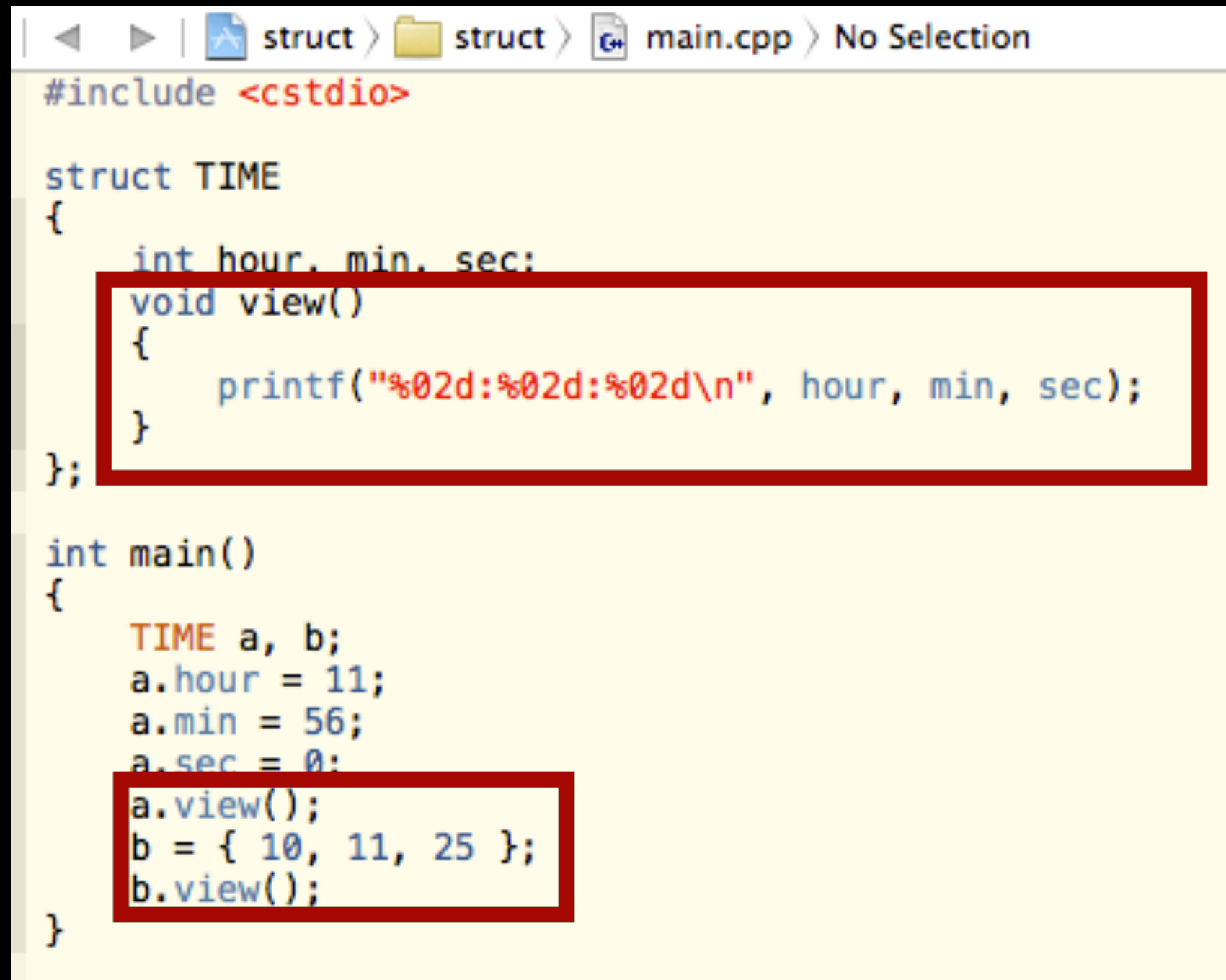
```
struct > struct > main.cpp > main()
#include <stdio>

struct TIME
{
    int hour, min, sec;
};

int main()
{
    TIME a;
    a.hour = 11;
    a.min = 56;
    a.sec = 0;
    printf("%02d:%02d:%02d\n", a.hour, a.min, a.sec);
}
```

- a라는 TIME형 변수에 각각의 값을 입력하고, 출력한다.
- 출력은 TIME형의 변수에 기본적으로 필요한 기능이라고 할 수 있다.

Method의 선언 및 활용



The image shows a code editor window with a file named 'main.cpp'. The code defines a struct 'TIME' with three integer members: 'hour', 'min', and 'sec'. A method 'view()' is declared within the struct, which uses 'printf' to display the values. In the 'main()' function, two 'TIME' objects, 'a' and 'b', are created. Object 'a' is initialized with values 11, 56, and 0. Object 'b' is initialized with values 10, 11, and 25. The 'view()' method is called on both objects using the dot operator. Two red rectangular boxes highlight the 'view()' method definition inside the struct and the calls to 'a.view()' and 'b.view()' in the main function.

```
< struct > struct > main.cpp > No Selection
#include <stdio>

struct TIME
{
    int hour, min, sec;
    void view()
    {
        printf("%02d:%02d:%02d\n", hour, min, sec);
    }
};

int main()
{
    TIME a, b;
    a.hour = 11;
    a.min = 56;
    a.sec = 0;
    a.view();
    b = { 10, 11, 25 };
    b.view();
}
```

- view라는 method를 선언하고 정의
- dot(.)연산자를 이용하여 method를 실행

구조체 초기화

```
struct > struct > main.cpp > main()
#include <stdio>

struct TIME
{
    int hour, min, sec;
    void view()
    {
        printf("%02d:%02d:%02d\n", hour, min, sec);
    }
};

int main()
{
    TIME a, b;
    a.view();
}
```

1606416448:32767:1606422622
Program ended with exit code: 0

- 구조체의 member들은 선언시 초기화를 지원하지 않는다.
- 즉 `int hour = 0, min = 0, sec = 0` 과 같이 선언할 수 없다.

구조체 생성자

```
struct > struct > main.cpp > main()
#include <stdio>

struct TIME
{
    int hour, min, sec;
    TIME()
    {
        hour = min = sec = 0;
    }
    void view()
    {
        printf("%02d:%02d:%02d\n", hour, min, sec);
    }
};

int main()
{
    TIME a, b;
    a.view();
    b.view();
}
```

```
00:00:00
00:00:00
Program ended with exit code: 0
```

- 생성자는 구조체의 이름과 같은 method이다.
- 구조체가 선언될 때, 자동으로 생성자 method가 실행된다.

method의 overloading

```
struct > struct > main.cpp > No Selection
#include <stdio>

struct TIME
{
    int hour, min, sec;
    TIME()
    {
        hour = min = sec = 0;
    }
    TIME(int h, int m, int s)
    {
        hour = h, min = m, sec = s;
    }
    void view()
    {
        printf("%02d:%02d:%02d\n",
};

int main()
{
    TIME a, b(12,11,0);
    a.view();
    b.view();
}
```

```
00:00:00
12:11:00
Program ended with exit code: 0
```

- 같은 이름으로 다른 파라미터를 가지는 함수를 선언하고 사용할 수 있다.
- 이러한 기능을 연산자, 함수의 overloading라 한다. 이는 C++에서 지원한다.

TIME의 덧셈 연산

```
struct > struct > main.cpp > No Selection
#include <stdio>

struct TIME
{
    int hour, min, sec;
    TIME()
    {
        hour = min = sec = 0;
    }
    TIME(int h, int m, int s)
    {
        hour = h, min = m, sec = s;
    }
    TIME add(TIME x)
    {
        TIME y;
        y.hour = hour + x.hour;
        y.min = min + x.min;
        y.sec = sec + x.sec;
        return y;
    }
    void view()
    {
        printf("%02d:%02d:%02d\n", hour, min, sec);
    }
};

int main()
{
    TIME a(5, 10, 22), b(12, 11, 0), c;
    c = a.add(b);
    c.view();
}
```

17:21:22

Program ended with exit code: 0

- add라는 함수를 이용하여 TIME의 덧셈을 구현
- 하지만 이 연산은 오류를 내포하고 있으므로, 오류를 없앨 수 있도록 수정하는 것은 스스로 해보기

operator을 이용한 덧셈 기호 적용

```
TIME operator+(TIME x)
```

```
{  
    TIME y;  
    y.hour = hour + x.hour;  
    y.min = min + x.min;  
    y.sec = sec + x.sec;  
    return y;  
}
```

```
int main()  
{  
    TIME a(5,10,22), b(12,11,0), c;  
    c = a + b;  
    c.view();  
}
```

17:21:22
Program ended with exit code: 0

- 실제 + 기호를 나타내는 method인 operator+ 를 이용
- 실제 +의 의미를 다시 정의하므로 이를 연산자의 재정의(overriding)라고 한다.
- overloading와 overriding의 의미를 잘 구분할 수 있도록 하자.

TIME 구조체를 이용하여 문제 풀기

★ Prob No.0265 : 오븐 시계 [CH04.2.Competition(KOI12Es-R)] ★

Time Limit(Test case) : 1000 (ms)

Total Submit : 376 Accepted : 166

The Champion of this Problem (C++) : **gs12016** - 0ms / 94byte

My Best Submission (C++) : **N/A**

Background

KOI 전자에서는 건강에 좋고 맛있는 훈제오리구이 요리를 간편하게 만드는 인공지능 오븐을 개발하려고 한다.

인공지능 오븐을 사용하는 방법은 적당한 양의 오리 훈제 재료를 인공지능 오븐에 넣으면 된다. 그러면 인공지능 오븐은 오븐구이가 끝나는 시간을 분 단위로 자동적으로 계산한다.

또한, KOI 전자의 인공지능 오븐 앞면에는 사용자에게 훈제오리구이 요리가 끝나는 시각을 알려 주는 디지털 시계가 있다.

훈제오리구이를 시작하는 시각과 오븐구이를 하는 데 필요한 시간이 분단위로 주어졌을 때, 오븐구이가 끝나는 시각을 계산하는 프로그램을 작성하시오.

Input

입력파일의 첫 째 줄에는 현재 시각이 나온다. 현재 시각은 시 A와 분 B가 정수로 빈칸을 사이에 두고 순서대로 주어진다.

두 번째 줄에는 요리하는 데 필요한 시간 C가 분 단위로 주어진다.

단, A는 23이하의 0을 포함한 자연수, B는 59이하의 0을 포함한 자연수, C는 1000이하의 0을 포함한 자연수이다.

Output

첫째 줄에 종료되는 시각의 시와 분을 공백을 사이에 두고 출력한다.

(단, 시는 0부터 23까지의 정수, 분은 0부터 59까지의 정수이다. 디지털 시계는 23시 59분에서 1분이 지나면 0시 0분이 된다.)

IO Example

입력

14 30

20

출력

14 50

큰 수의 덧셈을 구하는 프로그램도 작성해보자.

`struct BIG_NUM`

을 정의하고, +를 overriding하여 구현해보자.

- 구조체의 함수를 method라고 한다.
- 구조체와 이름이 같은 method를 생성자라고 한다.
- C++ 함수는 overload를 지원한다.
- C++의 operator을 이용하여 연산자를 overriding하여 사용할 수 있다.